ANALISIS KINERJA PERKALIAN MATRIKS PARALEL MENGGUNAKAN METRIK ISOEFISIENSI

Maria A. Kartawidjaja[1]

[1]*Jurusan Teknik Elektro*
Universitas Katolik Indonesia Atma Jaya
Jakarta
maria.kw@atmajaya.ac.id

ABSTRACT

In solving a complex problem is often less adequate sequential computing, especially in terms of execution time is very long, and thus become not fisibel. An interesting alternative to overcome the long computational time is the use of parallel computing. Kompuasi parallel performance can be measured by various metrics, one of them is isoefisiensi. With predictable isoefisiensi scalability of a parallel system. As a test program used in this study and the vector matrix multiplication, the matrix is partitioned following the chessboard pattern. Test program is executed on a computer with hypercube architecture. From the test results can be concluded that the parallel system is scalable because it requires a small increase in weight along with the increase in the number of processors.

Keywords: *isoefisiensi, performance, parallel, scalable*

PENDAHULUAN

Berbeda dengan kinerja komputasi sekuensial yang melulu ditentukan oleh algoritma yang digunakan, kinerja suatu komputasi paralel bergantung juga pada jumlah prosesor yang digunakan, pembagian beban pada masing-masing prosesor, komunikasi antar-prosesor, dan *overhead* lain yang timbul karena proses paralelisasi. Suatu algoritma yang memberikan kinerja yang baik pada suatu lingkup pemrograman paralel bisa saja memberikan kinerja yang buruk pada lingkup pemrograman paralel yang berbeda. Karena itu, evaluasi suatu algoritma paralel harus dilakukan secara komprehensif, misalnya dengan melakukan analisis skalabilitas suatu sistem paralel.

Walaupun tidak ada definisi yang baku untuk skalabilitas, secara umum skalabilitas menunjukkan kemampuan suatu sistem paralel untuk mempertahankan kinerjanya berdasarkan suatu metrik tertentu, seiring dengan bertambahnya jumlah prosesor. Dalam artikel ini metrik kinerja yang digunakan adalah isoefisiensi.

ISOEFISIENSI

Ada sejumlah metrik yang dapat digunakan untuk mengukur kinerja suatu sistem paralel. Metrik yang sering digunakan adalah peningkatan kecepatan atau yang lazim disebut sebagai *speedup* dan efisiensi prosesor.

Speedup adalah perbandingan antara waktu eksekusi algoritma sekuensial tercepat dengan waktu eksekusi algoritma paralel, atau waktu eksekusi algoritma paralel pada satu prosesor dengan waktu eksekusi algoritma tersebut pada sejumlah prosesor. Secara matematis *speedup* dinyatakan dengan Persamaan (1).

$$S_p = \frac{T_1}{T_p} \tag{1}$$

Speedup pada satu prosesor adalah sama dengan satu, dan *speedup* pada *p* prosesor bernilai $1 \le Sp \le p$. Secara ideal *speedup* meningkat sebanding dengan bertambahnya jumlah prosesor. Jadi jika digunakan *p* prosesor, *speedup* idealnya adalah *p*. Dalam beberapa kasus dapat terjadi *superlinear speedup* ($Sp > p$). *Superlinear speedup* tidak mencerminkan peningkatan kinerja yang sebenarnya, karena peningkatan kecepatan ini diakibatkan oleh fitur unik dari arsitektur paralel, misalnya ukuran *cache* yang lebih besar pada lingkup pemrograman paralel dibandingkan dengan ukuran *cache* pada lingkup pemrograman sekuensial [1, 2]. Dalam artikel ini tidak dibahas *superlinear speedup*.

Efisiensi merupakan suatu ukuran kinerja yang sangat erat hubungannya dengan *speedup*. Secara matematis efisiensi dinyatakan dengan

$$E = \frac{Sp}{p} \tag{2}$$

,dengan kisaran nilai antara $\frac{1}{p} \le E \le 1$.

Speedup dan efisiensi merupakan fungsi dari ukuran data dan jumlah prosesor. Umumnya *speedup* tidak akan meningkat secara linear dengan meningkatnya jumlah prosesor, melainkan cenderung mencapai suatu titik jenuh.

Dengan perkataan lain, efisiensi akan menurun jika jumlah prosesor meningkat. Nilai *speedup* dan efisiensi yang tidak ideal ini dikarenakan adanya *overhead* pada sistem paralel, misalnya komputasi tambahan yang hanya dibutuhkan pada sistem paralel, komunikasi antar-prosesor, dan proses sinkronisasi. Hal ini berlaku untuk semua sistem paralel, dan gejala saturasi dari *speedup* dan efisiensi ini mengikuti Hukum Amdahl [4].

Namun, *speedup* dan efisiensi akan meningkat jika ukuran data juga ditingkatkan, hal ini sesuai dengan Hukum Gustafson [5]. Jika peningkatan jumlah prosesor akan menurunkan efisiensi, dan peningkatan ukuran data akan meningkatkan efisiensi, berarti efisiensi dapat dibuat konstan jika jumlah prosesor dan ukuran data sama-sama ditingkatkan. Efisiensi yang konstan ini disebut sebagai isoefisiensi.

Kinerja suatu sistem paralel dapat dianalisis dengan menggunakan fungsi isoefisiensi. Sistem paralel yang memiliki fungsi isoefisiensi sebesar $\Theta(p)$ dikatakan sebagai suatu sistem paralel yang *scalable* (*scalable parallel system*).

SCALABLE PARALLEL SYSTEM

Untuk memperoleh efisiensi yang konstan, ukuran data perlu ditingkatkan seiring dengan meningkatnya jumlah

prosesor. Peningkatan ukuran data akan berbeda untuk sistem paralel yang berbeda, dan peningkatan ini dipengaruhi oleh berbagai faktor, seperti pendistribusian data antarprosesor dan kanal komunikasi pada suatu sistem paralel.

Skalabilitas sistem paralel dapat dianalisis dengan menggunakan suatu fungsi yang menghubungkan ukuran data dengan *overhead* dari sistem paralel. Fungsi ini dinamakan fungsi isoefisiensi. Suatu sistem paralel yang *scalable* adalah suatu sistem paralel yang mampu mempertahankan kinerjanya berdasarkan suatu metrik, seiring dengan peningkatan jumlah prosesor.

Jika T_1 dan T_p masing-masing menyatakan waktu eksekusi pada satu dan p prosesor, maka *overhead* sistem paralel dapat dinyatakan dengan [6]

$$T_o = pT_p - T_1 \quad (3)$$

atau

$$T_p = \frac{T_1 + T_o}{p} \quad (4)$$

Karena waktu eksekusi pada satu prosesor hanya ditentukan oleh beban kerja pada prosesor, maka $T_1 = W$, sehingga Persamaan (4) dapat ditulis sebagai

$$T_p = \frac{W + T_O}{p} \quad (5)$$

Dengan mensubstitusikan Persamaan (5) ke Persamaan (1) dan Persamaan (2), diperoleh

$$S = \frac{W \times p}{W + T_O} \quad (6)$$

$$E = \frac{1}{1 + T_O / W} \quad (7)$$

dan

Dengan mendefinisikan suatu konstanta $K = E/(1-E)$, maka Persamaan (8) dapat dinyatakan dengan [6]

$$W = KT_O \quad (9)$$

Persamaan (9) merupaka fungsi yang menunjukkan seberapa besar peningkatan beban kerja W yang dibutuhkan seiring dengan peningkatan jumlah prosesor untuk mempertahankan efisiensi sistem. Fungsi ini disebut sebagai fungsi isoefisiensi. Suatu fungsi isoefisiensi yang kecil menunjukkan bahwa untuk mempertahankan efisiensi sistem seiring dengan bertambahnya prosesor, hanya dibutuhkan sedikit peningkatan beban kerja, atau dengan perkataan lain sistem tersebut adalah suatu *scalable parallel system*.

JARINGAN INTERKONEKSI

Secara umum ada dua macam model pemrograman dengan arsitektur berbeda pada *platform* komputasi paralel, yaitu model memori bersama (*shared memory*) dan model pengiriman pesan (*message-passing*) [7]. Pada model memori bersama, memori dapat diakses oleh setiap prosesor sedangkan pada model pengiriman pesan, masing-masing prosesor memiliki memori lokal yang terpisah. Memori lokal ini hanya dapat diakses oleh prosesor yang bersangkutan. Pada arsitektur dengan model pengiriman pesan, jaringan interkoneksi antara prosesor umumnya berupa jaringan statis.

MODEL JARINGAN STATIS

Karakteristik jaringan yang digunakan untuk menghubungkan prosesor-prosesor pada suatu sistem paralel sangat mempengaruhi waktu komunikasi dari sistem paralel. Ada beberapa model jaringan yang umum digunakan, seperti jaringan bintang, jaringan linear (*ring*), jaringan *mesh*, jaringan pohon, dan jaringan *hypercube*. Pada penelitian ini digunakan komputer dengan jaringan interkoneksi berbentuk *hypercube*.

Besarnya waktu untuk pengiriman pesan antar-prosesor kecuali ditentukan oleh besarnya waktu *startup* suatu pengiriman dan kecepatan transfer, juga ditentukan oleh diameter dari jaringan interkoneksi. Diameter di sini menyatakan jarak maksimum antara dua prosesor dalam jaringan. Jika p menyatakan jumlah prosesor dalam suatu jaringan, maka diameter suatu jaringan linear adalah $\lfloor p/2 \rfloor$, diameter jaringan *mesh* adalah $2\lfloor \sqrt{p} - 1 \rfloor$ (*wraparound mesh*) dan $2\lfloor \sqrt{p}/2 \rfloor$ (*non-wraparound mes*h), dan diameter jaringan *hypercube* adalah *log p* [6].

MODEL WAKTU KOMUNIKASI

Pengiriman pesan yang paling sederhana adalah pengiriman pesan yang berlangsung antara dua prosesor. Jika t_s dan t_w masing-masing merepresentasikan waktu *startup* pengiriman dan kecepatan transfer pesan, maka waktu komunikasi antara dua prosesor dapat dinyatakan dengan

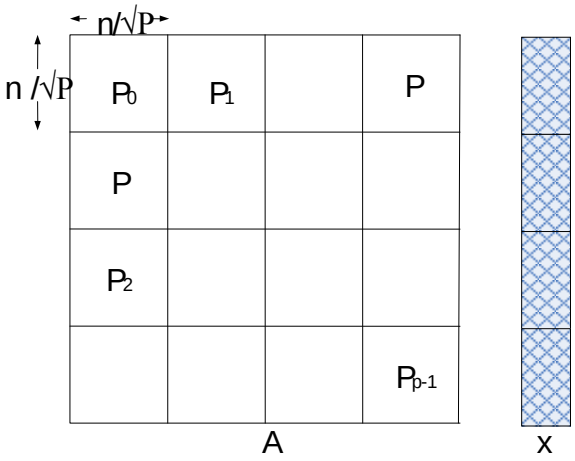
$$t_{comm} = t_s + mt_w l \quad (10)$$

Dengan m adalah panjang pesan dan l adalah panjang lintasan yang dilewati pesan untuk mencapai prosesor target.

Selain waktu komunikasi antara dua prosesor, juga terdapat empat jenis komunikasi lain, yaitu *single-node broadcast*, *multinode broadcast*, *single-node scatter*, dan *total exchange* [6]. Komunikasi *single-node broadcast* terjadi jika satu prosesor mengirim pesan yang sama secara serempak ke sejumlah prosesor; komunikasi *multinode broadcast* adalah bentuk umum dari *single-node broadcast,* dengan sejumlah prosesor secara serempak menginisialisasi pengiriman pesan. Satu prosesor mengirim pesan yang sama ke sejumlah prosesor, namun pesan yang dikirim oleh prosesor lain bisa saja berbeda. Komunikasi *one-to-all personalized communication* terjadi jika satu prosesor mengirim pesan yang berbeda secara serempak ke sejumlah prosesor; dan komunikasi *all-to-all personalized communication* terjadi jika sejumlah prosesor mengirim pesan yang berbeda untuk sejumlah prosesor lain. Keempat komunikasi berlaku baik untuk pengiriman ataupun penerimaan pesan. Ini berarti komunikasi tersebut bersifat dualistis. Komunikasi dualistis dari *single-node broadcast* adalah *single-node accumulation*, dari *multinode broadcast* adalah *multinode accumulation*, dan dari *single-node scatter* adalah *single-node gather*.

HASIL PENGUJIAN DAN ANALISIS

Sebagai problem uji digunakan program perkalian matriks dengan vektor, karena operasi ini banyak digunakan untuk penyelesaian masalah komputasi dalam pemodelan fenomena alam.



■ **Gambar 1.** Dekomposisi matriks A dan vektor x

HASIL PENGUJIAN

Komputasi paralel dilakukan untuk matriks dengan dekomposisi data mengikuti pola papan catur seperti tampak pada Gambar 1. Dengan demikian prosesor yang digunakan di sini haruslah dalam jumlah kuadrat, misalnya satu (1^2), empat (2^2), sembilan (3^2) prosesor, dan seterusnya. Dari hasil pengujian diperoleh waktu eksekusi seperti yang ditampilkan padaTabel 1.

Dari Tabel 1 dapat dihitung *speedup* menggunakan Persamaan (1), dan efisiensi dengan Persamaan (2). Hasil perhitungan efisiensi ditampilkan pada Tabel 2.

ANALISIS SKALABILITAS

Jika unit waktu yang dibutuhkan untuk melakukan operasi penjumlahan diasumsikan sama dengan unit waktu untuk melakukan operasi perkalian dan dinyatakan dengan t_c, maka waktu komputasi perkalian matriks dengan vektor secara sekuensial tampak pada persamaan (11).

$$T_{komp}^{sek} = n(2n-1)\times t_c \approx 2n^2 \times t_c \qquad (11)$$

Pada awalnya satu posesor yang merupakan *root processor*, mengirimkan subblok matriks A dengan ukuran $n/\sqrt{p} \times n/\sqrt{p}$, dan vektor x dengan ukuran $n/\sqrt{p}$ ke semua prosesor lain. Ini berarti dibutuhkan suatu komunikasi *one-to-all personalized communication* untuk mengirim suatu paket pesan yang terdiri dari suatu subblok matriks A dan suatu potongan vektor x. Komunikasi ini dinyatakan dengan rumus seperti persamaan (12).

$$t_{com1} = \left(t_S + t_W\left(\frac{n^2}{\sqrt{p}} + \frac{n}{\sqrt{p}}\right)\right)\log_2\left(\sqrt{p}\right)$$
$$t_{com1} \approx \left(t_S + t_W\frac{n^2}{\sqrt{p}}\right)\log_2\left(\sqrt{p}\right) \qquad (12)$$

Untuk dapat melakukan operasi perkalian matriks dengan vektor pada setiap prosesor, maka masing-masing prosesor harus memiliki seluruh potongan vektor x. Oleh karena potongan vektor x hanya dimiliki oleh prosesor tertentu, masing-masing prosesor akan mengirim potongan vektor x yang dimilikinya ke prosesor lain. Hal ini berarti diperlukan suatu komunikasi *all-to-all broadcast* seperti yang tampak pada Gambar 2.

■ **Tabel 1.** Waktu eksekusi perkalian matriks dengan vektor

Dimensi matriks	Waktu eksekusi (milidetik)				
n	T_1	T_4	T_9	T_{16}	T_{25}
$1x10^4$	10,86	6,98	5,15	3,80	2,76
$2x10^4$	17,53	8,29	5,42	4,28	2,88
$3x10^4$	48,18	16,29	9,53	5,70	4,72
$4x10^4$	88,20	25,72	12,74	7,57	6,24
$5x10^4$	135,30	37,51	17,78	10,52	8,50
$6x10^4$	194,80	52,68	24,00	13,92	10,69
$7x10^4$	267,29	68,69	31,23	18,23	13,54

■ **Tabel 2.** Efisiensi perkalian matriks dengan vektor

Dimensi matriks	Efisiensi			
n	E_4	E_9	E_{16}	E_{25}
$1x10^4$	0,39	0,23	0,18	0,16
$2x10^4$	0,53	0,36	0,26	0,24
$3x10^4$	0,74	0,56	0,53	0,41
$4x10^4$	0,86	0,77	0,73	0,57
$5x10^4$	0,90	0,85	0,80	0,64
$6x10^4$	0,92	0,90	0,87	0,73
$7x10^4$	0,97	0,95	0,92	0,79

Waktu komunikasi yang dibutuhkan untuk pengiriman potongan-potongan vektor x berukuran masing-masing $n/\sqrt{p}$ dengan teknik *all-to-all broadcast* pada arsitektur *hypercube* adalah:

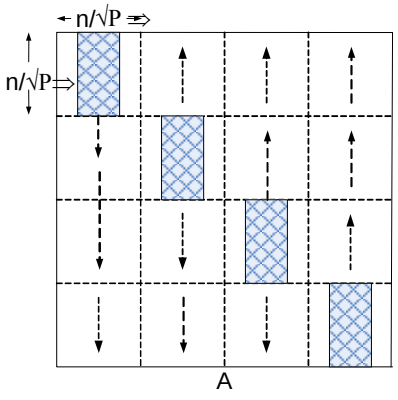
$$t_{comm} = \left(t_S + t_W\frac{n}{\sqrt{p}}\right)\log_2\left(\sqrt{p}\right) \qquad (13)$$

Sesudah itu setiap prosesor akan melakukan komputasi lokal dengan waktu pada Persamaan (14).

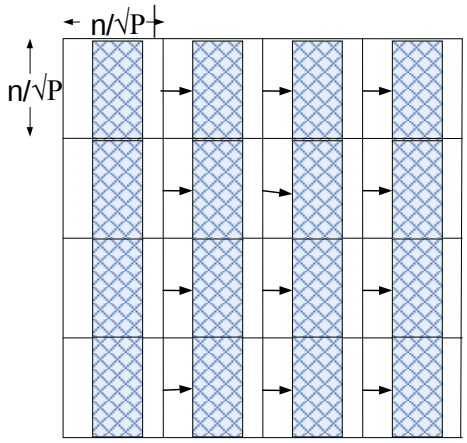
$$T_{komp}^{par} \approx \frac{2n^2}{p} \qquad (14)$$

Hasil komputasi dari masing-masing prosesor yang berupa vektor dengan ukuran $n/\sqrt{p}$ dikumpulkan, yang berarti dibutuhkan suatu komunikasi *single-node accumulation* yang besarnya sama dengan komunikasi *all-to-all broadcast* pada Persamaan (13) di atas. Komunikasi ini ditunjukkan pada Gambar 3.

Dengan demikian waktu eksekusi paralel tampak pada persamaan (15).



■ **Gambar 2.** Komunikasi *all-to-all broad-cast*



■ **Gambar 3.** Komunikasi *single-node accumulation*

$$T_{total}^{par} \approx \frac{2n^2}{p} + 3\left(t_S + t_W \frac{n^2}{\sqrt{p}}\right)\log_2\left(\sqrt{p}\right) \quad (15)$$

Berdasarkan Persamaan (3) *overhead* sistem paralel tampak pada persamaan (16).

$$T_O = 3\left(t_S \sqrt{p}\log_2\sqrt{p} + t_W n^2 \log_2\sqrt{p}\right) \quad (16)$$

Analisis kinerja dengan metrik isoefisiensi dilakukan untuk masing-masing komponen *overhead* pada Persamaan (16) dengan menggunakan Persamaan (9). Komponen *overhead* dengan fungsi isoefisiensi yang bernilai paling besar menentukan skalabilitas keseluruhan sistem paralel. Dari Persamaan (16) diperoleh dua fungsi isoefisiensi berikut:

$$W(t_S) = 3Kt_S\sqrt{p}\log_2\left(\sqrt{p}\right) \quad (16a)$$

$$W(t_W) = n^2 = 3Kt_W n^2 \log_2\left(\sqrt{p}\right) \quad (16b)$$

Dari Persamaan (16) tampak bahwa *overhead* melulu ditentukan oleh parameter t_S dengan kompleksitas pada Persamaan (17).

$$W(t_S) = O\left(\sqrt{p}\log_2\left(\sqrt{p}\right)\right) \quad (17)$$

Fungsi ini menyatakan bahwa untuk mendapatkan efisiensi yang konstan jika jumlah prosesor ditingkatkan dari p menjadi $\tilde{p}$, maka ukuran data perlu ditingkatkan dengan faktor $\frac{\sqrt{\tilde{p}}\log_2\left(\sqrt{\tilde{p}}\right)}{\sqrt{p}\log_2\left(\sqrt{p}\right)}$.

Dari Tabel 2 tampak bahwa efisiensi pada empat prosesor untuk ukuran data $5x10^4$ adalah 0,90. Supaya efisiensi ini konstan, maka jika digunakan sembilan prosesor, ukuran data perlu ditingkatkan sebesar

$$\frac{\sqrt{9}\log_2\left(\sqrt{9}\right)}{\sqrt{4}\log_2\left(\sqrt{4}\right)} \times 5\times10^4 \approx 5{,}94\times10^4 \quad (18)$$

Dari Tabel 2 tampak efisiensi sebesar 0,90 dengan sembilan prosesor diperoleh untuk ukuran data $6x10^4$. Untuk penggunaan prosesor yang lebih banyak lagi, misalnya 16 prosesor, maka untuk mencapai efisiensi yang kira-kira sama dibutuhkan peningkatan data sebesar

$$\frac{\sqrt{16}\log_2\left(\sqrt{16}\right)}{\sqrt{4}\log_2\left(\sqrt{4}\right)} \times 5\times10^4 \approx 10\times10^4 \quad (19)$$

sedangkan dari Tabel 2 tampak ukuran data yang dibutuhkan untuk mencapai efisiensi 0,92 adalah $7x10^4$. Perbedaan ini bisa saja terjadi mengingat dalam analisis isoefisiensi tidaklah diperhitungkan kemungkinan adanya tumpang-tindih (*overlap*) antara waktu komputasi dan waktu komunikasi, padahal dalam komputasi paralel justru diusahakan terjadi proses tumpang-tindih guna menurunkan waktu eksekusi yang akan berdampak pada peningkatan kinerja komputasi. Berdasarkan hasil pengujian tampak bahwa sistem paralel ini adalah suatu sistem paralel yang *scalable*.

KESIMPULAN

Dari hasil pengujian dan analisis dapat ditarik kesimpulan bahwa sistem paralel untuk perkalian matriks dan vektor dengan matriks yang dipartisi menurut pola papan catur merupakan suatu sistem paralel yang *scalable*, karena hanya membutuhkan peningkatan beban yang kecil seiring dengan peningkatan jumlah prosesor.

DAFTAR PUSTAKA

[1]. Freeman, T. L. and Phillips, C. 1992. *Parallel Numerical Algorithms*. New York: Prentice-Hall, Inc.
[2]. Foster, I. 1995. Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Massachuttes: Addison-Wesley Publi-shing Company.
[3]. Wilkinson, B. dan Allen, M. 1999. Parallel Programming. New Jersey: Prentice-Hall, Inc.
[4]. Amdahl, G.M. 1967. Validity of The Single Processor Approach to Achieving Large-Scale Computing Capability. Proc. AFIPS Conference, pp 483-485
[5]. Gustafson, J.L. Reevaluating Amdahl's Law. Comm. ACM, 31 (5): 532-533.
[6]. Kumar, V. et al. 1994. Introduction to Parallel Computing: Design and Analysis of Algorithms. 1994. CA: Benjamin/Cummings Publishing Company, Inc.
[7]. Sun, X. H. and J. Zhu. 1995. Performance Considerations of Shared Virtual Memory Machines. *IEEE Transactions on Parallel and Distributed Systems*, 6(11):1185-1194